Conditional Bigraphs

Blair Archibald(✉), Muffy Calder, and Michele Sevegnani

School of Computing Science, University of Glasgow, Glasgow, UK
{blair.archibald,muffy.calder,michele.sevegnani}@glasgow.ac.uk

Abstract. Bigraphs are a universal graph based model, designed for analysing reactive systems that include spatial and non-spatial (*e.g.* communication) relationships. Bigraphs evolve over time using a rewriting framework that finds instances of a (sub)-bigraph, and substitutes a new bigraph. In standard bigraphs, the applicability of a rewrite rule is determined completely by a local match and does not allow any non-local reasoning, *i.e.* contextual conditions. We introduce conditional bigraphs that add conditions to rules and show how these fit into the matching framework for standard bigraphs. An implementation is provided, along with a set of examples. Finally, we discuss the limits of application conditions within the existing matching framework and present ways to extend the range of conditions that may be expressed.

Keywords: Bigraphs · Bigraphical reactive systems · Application conditions · Conditional rewriting

1 Introduction

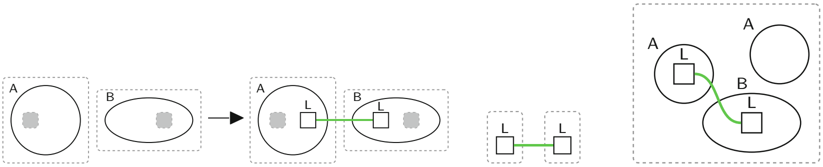
Bigraphs are a universal mathematical model, introduced by Milner [15], for representing spatial and non-spatial relationships of physical or virtual entities. They have been applied to a wide range of systems including: mixed-reality systems [4], networking [6], Iot [2], security of cyber-physical systems [1], and biology [13].

Bigraphical reactive systems (BRS) augment bigraphs with a rewriting theory that allows models to evolve over time. The rewrite theory consists of a set of reaction rules $L \longrightarrow\!\!\!\!\blacktriangleright R$ that finds an *occurrence* of L in a larger bigraph B and replaces it with R. This form of rewriting only allows local reasoning through *matching* a pattern bigraph L exactly, but does not allow non-local reasoning that takes into account the context of a rule. We introduce *conditional* rules that use *application conditions* to specify contextual requirements within the rewrite system. Such conditional rules have proved invaluable in graph transformation systems [11] (GTS), a closely related formalism to bigraphs. However, it is important to note that although BRS and GTS are based on graph structures, the formalisms require completely different semantics for conditional rewriting. For example, in GTS there is a single context for the rules, whereas BRS feature a distinct *context* and a *parameter*, and so application conditions can be specified over either.

F. Gadducci and T. Kehrer (Eds.): ICGT 2020, LNCS 12150, pp. 3–19, 2020.
https://doi.org/10.1007/978-3-030-51372-6_1

A common requirement for conditional rules in BRS is to avoid the duplication of links between entities. As an example, consider the `createLink` rule shown in Fig. 1a (full details of this notation is in Sect. 2). Bigraphs consist of entities, A, B and L, shown as shapes that are related either by a nesting relationship – L *inside* A – or via the green hyperlinks. Sites (grey rectangles) represent parts of the system that have been abstracted away, *i.e.* other bigraphs may appear inside. Without an application condition, this rule allows *any* number of L-L links to be created between an A and a B; this is because the sites may contain any number of other entities, including existing L entities. If we wish to restrict to *single* L-L links between A-B pairs, we have to employ some sort of *tagging* scheme [6], often coupled with *rule priorities* [3], that can determine when a link does or does not exist. In practice, this requires an extra entity (for tagging) and additional (four) reaction rules. This inflates the model with non-domain specific rules, generates additional control-only steps in the resulting transition system, and, more importantly, obfuscates the purpose of the rule, which is to create non-duplicate links.

With conditional rules, we can achieve this goal in a *single* rewrite step. In the example, we create a conditional rule though the addition of a negative application condition, which is shown in Fig. 1b. This states that within the *parameter* of the rule, *i.e.* in the *sites*, we must not find an existing L-L link. If such a link is found then the rule does not apply. Consider application of the rule to the example bigraph in Fig. 1c. For the linked A, the existing L entities appear *inside* the two sites of the parameter (of the left-hand side of the rule). As the parameter negative condition forbids such a shape to appear in the left-hand side, no new link can be created. On the other hand, for the unlinked A, no L-L link is present in the parameter and so the negative condition is not satisfied, and a new link can be created.



(a) `createLink` rule (b) (c)

Fig. 1. Negative application condition to avoid duplicate L-L links. (b) Negative application condition for the parameter. Given the bigraph in (c), the rule does not apply for the already linked A and B but does for the unlinked A and B. (Color figure online)

In Sects. 3–5 we show that conditional rules are possible within Milner's original BRS formalism, and we give our implementation in BigraphER [19]. In Sect. 6 we reflect on the fact that, due to how bigraph matching is defined, conditional rules are limited in the conditions that can be expressed. We discuss these limitations, with examples, and indicate possible extensions that include matching on names and patterns, spatial logics to encode spatial context, and matching with sorting schemes, site numbering, and nested application conditions.

We make the following contributions:

- We extend the original formalism of bigraphs to support non-local application conditions for bigraphical reactive systems.
- To show application conditions are both implementable and useful, we implement application conditions in BigraphER [19] and provide example models that highlight common uses of application conditions in practice.
- We show the limits of such an approach, based strictly on the existing matching/decomposition of bigraphs, and highlight areas for future exploration.

2 Bigraphs

A bigraph consists of two orthogonal structures: a *place graph* that describes the *nesting* of entities, *e.g.* a Phone inside a Room, and *link graph* that provides non-local hyperlinks between entities, *e.g.* allowing Phone entities to communicate regardless of location. In standard bigraphs place graphs are forests, however here we use bigraphs with sharing [18], that has place graphs as directed acyclic graphs, allowing entities to have multiple parents. Bigraphs feature an equivalent algebraic and an intuitive diagrammatic form, and we use this diagrammatic form where possible.

An example bigraph is in Fig. 1c. We draw entities as different (colored) shapes, often omitting the label when possible. Containment illustrates the spatial nesting relationship, *e.g.* L is contained by A, while green hyperedges represent non-spatial connections. Entities have a fixed *arity* (number of links), *e.g.* L has arity 1, but links may be disconnected/closed.

Each place graph has m *regions*, shown as the dashed rectangles, and n *sites*, shown as filled dashed rectangles. Regions represent parallel parts of the system, and sites represent abstraction, *i.e.* an unspecified bigraph (including the empty bigraph) exists there. Similarly, link graphs have a (finite) set of inner names, *e.g.* $\{z\}$ and outer names, *e.g.* $\{x, y\}$. For example, in Fig. 2b, C has an inner name x, d has outername x, and id_I has both an inner and outer name x (where both x's are distinct).

Bigraphs are compositional structures, that is, we can build larger bigraphs from smaller bigraphs. Composition of bigraphs consists of placing regions in sites, and connecting inner and outer-faces on like-names.

Algebraically we describe bigraphs using their *interfaces*, *e.g.* $B : \langle n, X\rangle \rightarrow \langle m, Y\rangle$, or more succinctly $B : I \rightarrow J$, where n is the number of sites, m number of regions, X a set of inner names, and Y a set of outer names. Composition of bigraphs is defined when the interfaces match, *i.e.* $B_1 \circ B_0$ is defined for $B_0 : I \rightarrow J$ and $B_1 : J \rightarrow K$. We use ϵ to refer to the empty interface $\langle 0, \emptyset\rangle$, and call bigraphs of the form $\epsilon \rightarrow I$ *ground*, *i.e.* bigraphs with no sites and no inner names. Figure 1a is non-ground as it contains two sites, while Fig. 1c is ground as it contains no sites or inner names.

We let $\mathsf{id}_I : \mathsf{id}_I : \langle m, X\rangle \rightarrow \langle m, X\rangle$ be the identity bigraph over an interface $I : \langle m, X\rangle$. id_I maps names in X to themselves and places m sites in m regions. This bigraph is particularly important for matching as it allows names and entities to move between the context and parameter of a match. With these definitions, bigraphs form a pre-category[1] with objects as interfaces and arrows as bigraphs.

While composition combines bigraphs *vertically*, we can also combine bigraphs *horizontally* through the tensor product $\otimes$. This tensor product extends both the sites/regions and name sets for the interfaces. For example, given $A : \langle 0, \{x\}\rangle \rightarrow \langle 1, \{y\}\rangle$ and $B : \langle 1, \emptyset\rangle \rightarrow \langle 2, \{z\}\rangle$, we can construct a new bigraph $A \otimes B : \langle 1, \{x\}\rangle \rightarrow \langle 3, \{y, z\}\rangle$ Note that $\otimes$ is only defined when the sets of interface names are disjoint.

Notation. When referring to a ground bigraph we use lower-case letters, while general bigraphs, that may or may not be ground, are denoted in upper-case. Where the identity of an interface is not required we use $\cdot$ as a placeholder for $I, J, \ldots$

2.1 Bigraphical Reactive Systems

Bigraphical reactive systems (BRS) equip bigraphs with a rewriting theory that allows models to evolve over time. Intuitively, applying a reaction rule $L \longrightarrow R$ to bigraph B finds an *occurrence* of L in B (if one exists) and replaces it with R to create B'. Most often we rewrite over *ground* bigraphs as these represent fully formed models, *e.g.* without holes/sites. Here we give the most general definitions possible, *i.e.* for arbitrary (including ground) bigraphs B, and specialise to ground bigraphs when necessary.

We work with a restricted version of reaction rules that are "well-behaved" where L is *solid*[2]. Solid bigraphs were introduced by Krivine *et al.* [13] to count unique occurrences for stochastic BRS.

Definition 1 (solid). *A bigraph is* solid *if:*

- *All roots contain at least one node, and all outer names are connected to at least one edge.*
- *No two sites or inner names are siblings*
- *No site has a root as a parent*
- *No outer name is linked to an inner name.*

Definition 2 (occurrence). *We say a bigraph P* occurs *in B, written $B \vDash P$, if there exists a decomposition $B = C \circ (P \otimes \mathsf{id}_I) \circ D$ for some context C and parameter D. That is, there is a match for P in B.*

Likewise, we say P does not occur in B, written $B \nvDash P$ if $\nexists C' \nexists D', B = C' \circ (P \otimes \mathsf{id}_I) \circ D'$. That is, there is no match for P in B.

[1] Bigraphs are not a full category as composition is not defined for non-disjoint *supports*. We do not discuss support here.

[2] The definition of solid for bigraphs with sharing differs slightly, see [17, Defn 3.6.1].

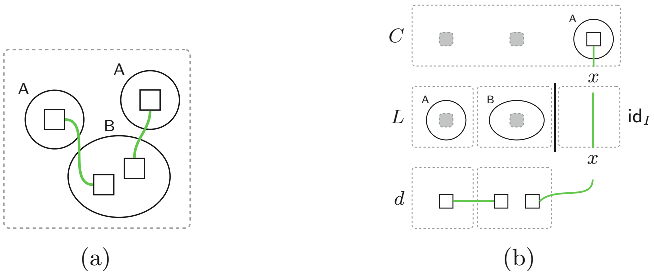


Fig. 2. Decomposition of ground bigraph $b = C \circ (L \otimes \mathsf{id}_I) \circ d$.

For a solid L, we gain the property that an occurrence $B \models L$ *uniquely identifies* a context C and parameter D.

We show graphically how occurrences are found in Fig. 2. Given the ground bigraph b shown in Fig. 2a, we show in Fig. 2b one (of the two possible) decompositions when matching against the rule given in Fig. 1a. The match L, by definition, has the same form as the left-hand-side of the reaction rule. The context C captures entities in the bigraph that do not lie within the match, while the parameter d, which must be ground as b is ground, provides the entities required to fill any sites/inner names in the match. To allow names (and entities in the case of sharing, *i.e.* for those sharing a parent in the match and context) to move between the parameter and the context, we allow an interface id_I next to the match. The use of id_I means the distinction between context and parameter is not always clear as both names and unmatched entities can move between the context and parameter as required, *e.g.* an entity from the context can appear in the parameter by extending id_I with an additional, trivial, region/site. Note that we only take L to be solid allowing these region/sites to be added as required.

Allowing entities to move between the context and parameter complicates the specification of application conditions that must determine if the condition is within the context of the parameter. We deal with this by forcing the parameter to be *minimal* such that it contains only entities that are within sites of the match, and all other entities move to the context. We note this is a choice and the theory also applies to systems that take the minimal context.

Definition 3 (reaction rule). *A* reaction rule R *is a pair of bigraphs,* $\mathsf{R} = (L, R)$*, defined over the same interface, and often written as* $L \longrightarrow R$*, with* L *solid. Applying a reaction rule consists of replacing an occurrence* $B \models L$*, in a given bigraph* B*, with* R*.*

Rewriting (over ground bigraphs) is shown graphically in the commuting diagram of Fig. 3. Given a ground bigraph a, we first find a decomposition $a = C \circ (L \otimes \mathsf{id}_I) \circ d$ and, should such a decomposition exist, rewrite it to obtain $a' = C \circ (R \otimes \mathsf{id}_I) \circ d$. Both the context C and parameter d are the same for both the left and right hand sides of the rewrite[3]. In Sect. 3 we show how checking application conditions corresponds to a further decomposition of C and d.

$$\begin{tikzcd}
 & & \cdot & & \\
\epsilon \arrow[r, "d"'] \arrow[urr, bend left, "a"] & \cdot \arrow[r, "L \otimes \mathsf{id}_I"'] & \cdot \arrow[u, "C"'] & \cdot \arrow[l, "R \otimes \mathsf{id}_I"] \arrow[ul, phantom, "\doteq"] & \epsilon \arrow[l, "d"] \arrow[ull, bend right, "a'"']
\end{tikzcd}$$

Fig. 3. Bigraph rewriting [15]. a rewrites to (a support equivalent) a' if there is a decomposition $a = C \circ (L \otimes \mathsf{id}_I) \circ d$ and rule $L \mathrel{-\!\!\blacktriangleright} R$.

While the rules are specified for *abstract* bigraphs, *i.e.* they match any entity of the correct type, rewriting itself works on *concrete* bigraphs where entities have distinct identifiers. In general, we may rewrite into any *support equivalent* a', notated $\doteq$ where support equivalence allows renaming of entity identifiers and link-names while keeping the structure intact. For the rest of the paper we assume support equivalence without explicitly stating it.

Given a set of reaction rules we construct a BRS as follows.

Definition 4 (bigraphical reactive system (BRS)). *A* bigraphical reactive system *consists of a set of* ground *bigraphs $\mathcal{B}$ and set of reaction rules $\mathcal{R}$, defined over $\mathcal{B}$, of the form $L \mathrel{-\!\!\blacktriangleright} R$. The reaction relation $\mathrel{-\!\!\triangleright}$ over ground bigraphs is the smallest such that $b \mathrel{-\!\!\triangleright} b'$ when $b = C \circ (L \otimes \mathsf{id}_I) \circ d$ and $b' = C \circ (R \otimes \mathsf{id}_I) \circ d$ for $b, b' \in \mathcal{B}$, reaction $(L \mathrel{-\!\!\blacktriangleright} R) \in \mathcal{R}$, context C, and parameter d.*

That is, our system consists all possible (ground) bigraphs closed under $\mathrel{-\!\!\triangleright}$.

3 Application Conditions for Bigraphs

We show how application conditions for bigraphs are instances of the bigraph matching problem. We begin by defining application conditions, which can be viewed as stand-alone instances of the left-hand-side of a rule.

Application conditions include information such as if they are positive or negative and if they apply to the context or parameter of a match.

Definition 5 (application condition). *An* application condition *is a tuple $\langle t, P, l\rangle$ where $t \in \{+, -\}$ is the type of application condition, either positive or negative, P is a (non-ground, not necessarily solid) constraint bigraph, and $l \in \{\uparrow, \downarrow\}$ determines if the condition is over the context ($\uparrow$) or parameter ($\downarrow$).*

[3] Bigraphs allow the use of an instantiation map η [15, Defn 8.3] that specifies a mapping of sites in the left-hand side to those in the right-hand site. We do not consider instantiation maps here.

Checking the conditions is a matching problem and no *rewriting* is performed on the context/parameter, *i.e.* we do not reduce constraints. Finally, we define conditional reaction rules.

Definition 6 (conditional reaction rule). *A* conditional reaction rule R *consists of a reaction rule* $\mathsf{R} : L \longrightarrow R$*, and a set of application conditions* $\mathcal{A}$*. Unconditional rules are those where* $\mathcal{A} = \emptyset$*.*

A rule $L \longrightarrow R$ *applies to* a *if there is a decomposition* $a = C \circ (L \otimes \mathsf{id}_I) \circ d$*, and*

$$\begin{aligned}
\forall \langle +, P, \uparrow \rangle \in \mathcal{A},\ C \vDash P \\
\forall \langle -, P, \uparrow \rangle \in \mathcal{A},\ C \nvDash P \\
\forall \langle +, P, \downarrow \rangle \in \mathcal{A},\ d \vDash P \\
\forall \langle -, P, \downarrow \rangle \in \mathcal{A},\ d \nvDash P
\end{aligned}$$

In a slight overload of notation, we use $C, d \vDash a$ *when an application condition* $a \in \mathcal{A}$ *is satisfied (positively or negatively) in context* C *and parameter* d*.*

We show graphically how application conditions of the form $\langle +, P, \uparrow \rangle$ and $\langle +, P', \downarrow \rangle$ are checked in Fig. 4. This diagram shows the left-hand side of Fig. 3, *i.e.* the decomposition of ground bigraph a, but with the context C and parameter d further decomposed. As C can be decomposed into three arrows and as P exists in the decomposition we know the application condition is met. The use of the additional parameter E allows for application conditions to have a different interface from the original match (shown here as an arbitrary $\cdot$). Without E all application conditions would be forced to have the same number of sites as regions of L, and matching outer/inner names, making it difficult to specify reusable conditions. Likewise we can check P' via the decomposition of d. For negative application conditions we instead show that no such decomposition exists for a given P, *i.e.* we cannot form the diagram in Fig. 4.

$$\begin{tikzcd}
\epsilon \arrow[r, "d'"'] \arrow[rrr, bend right, "d"'] \arrow[rrrrrrr, bend left, "a"] & \cdot \arrow[r, "(P' \otimes \mathsf{id}_J)"'] & \cdot \arrow[r, "C''"'] & \cdot \arrow[r, "(L \otimes \mathsf{id}_K)"'] & \cdot \arrow[r, "E"'] \arrow[rrr, bend right, "C"'] & \cdot \arrow[r, "(P \otimes \mathsf{id}_I)"'] & \cdot \arrow[r, "C'"'] & \cdot
\end{tikzcd}$$

Fig. 4. Context and parameter decomposition over left-hand side of Fig. 3 to check positive application conditions

When multiple application conditions are specified, we check that matches exist (or do not exist) separately for each condition. That is, we take the conjunction of the conditions. Importantly, we do not force the decompositions to cover a unique set of entities, and allow the same entity to be matched for multiple application conditions, *i.e.* conditions can overlap. We discuss non-overlapping rules as an extension to this approach in Sect. 6.4.

Definition 7 (conditional bigraphical reactive system). *A conditional BRS consists of a set of* ground *bigraphs $\mathcal{B}$ and set of conditional reaction rules $\mathcal{R}_c$ of the form $(L \longrightarrow\!\!\blacktriangleright R, \mathcal{A})$.*

The reaction relation $\longrightarrow\!\!\triangleright$ is the smallest such that $b \longrightarrow\!\!\triangleright b'$ when $b = C \circ (L \otimes \mathsf{id}_I) \circ d$ and $b' = C \circ (R \otimes \mathsf{id}_I) \circ d$, for $b, b' \in \mathcal{B}$, reaction $(L \longrightarrow\!\!\blacktriangleright R, \mathcal{A}) \in \mathcal{R}$, context C, and parameter d. ***Additionally,*** *$\forall a \in \mathcal{A}$, $C, d \vDash a$.*

When $\mathcal{A} = \emptyset$, a conditional BRS is a standard BRS as in Definition 4.

4 Implementation

We have implemented application conditions in BigraphER [19] an open-source framework for bigraphs[4]. BigraphER supports bigraphs with sharing [18], including an efficient matching algorithm based on SAT that we use to decompose the bigraph to check application condition predicates.

We show the example rule for Fig. 1 in the BigraphER language in Listing 1.1. The rules are written as before, with the addition of an `if` clause that allows application conditions to be specified as arbitrary bigraphs. The structure of the `conds` production has the following BNF specification and appears as an optional statement of any reaction rule; the production ⟨bigraph_exp⟩ parses an arbitrary bigraph expression. As is common in programming languages, we use `!` to represent negation, while `param` and `ctx` become reserved keywords that specify *where* we should search for the constraint specified by ⟨bigraph_exp⟩ – in the parameter (sites) or context respectively.

$$\begin{aligned}
\langle \text{place} \rangle &::= \texttt{param} \mid \texttt{ctx} \\
\langle \text{bang} \rangle &::= \texttt{!} \mid \epsilon \\
\langle \text{app\_cond} \rangle &::= \langle \text{bang} \rangle\ \langle \text{bigraph\_exp} \rangle\ \texttt{in}\ \langle \text{place} \rangle \\
\langle \text{app\_conds} \rangle &::= \langle \text{app\_cond} \rangle \mid \langle \text{app\_cond} \rangle\,\texttt{,}\ \langle \text{app\_conds} \rangle \\
\langle \text{conds} \rangle &::= \texttt{if}\ \langle \text{app\_conds} \rangle
\end{aligned}$$

5 Examples

We apply conditional rewriting to three typical examples: ensuring entities are unique, implementing a $\nexists$ operator and performing counting in multi-sets, and replacing priorities/control with conditionals.

[4] Available, along with the example models of Sect. 5, at www.dcs.gla.ac.uk/~michele/bigrapher.html.

Listing 1.1: Specifying application conditions in BigraphER.

```
# Example from Fig. 1

# control <name> = <arity>
ctrl A = 0; # Circle
ctrl B = 0; # Ellipse
ctrl L = 1; # Square

react createLink =
A.id || B.id --> /x (A.(L{x} | id) || B.(L{x} | id))
if
!(/y (L{y} || L{y})) in param;
```

5.1 Uniqueness of Entities

Many applications have constraints on the uniqueness of particular entities. For example, in a networking application *e.g.* in [20], we want to disallow two devices having the same MAC address.

In bigraphs there is no general method to declare an entity as *unique*. However, with application conditions we can check, before an entity is created, that no identical entity exists in either the context *or* the parameter. Since there is no way to create a duplicate, if we use conditional reaction rules with appropriate conditions to generate a model, this will ensure entities are unique.

As an example, consider the rule `createUnique` shown in Fig. 5. This rule allows an entity Unique to be created in a given Place so long as no other entity Unique already exists in the model – either in the same place, *i.e.* the parameter, or anywhere in the context.

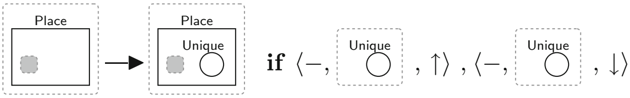
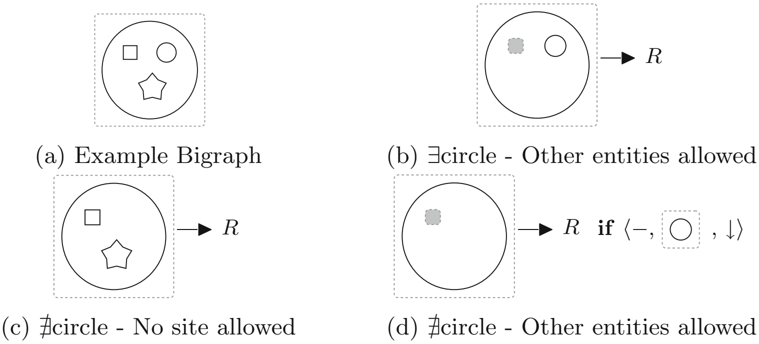


Fig. 5. `createUnique` – application conditions force uniqueness of entity Unique.

5.2 Non-existence and Counting in Multisets

Non-atomic entities in bigraphs can be considered multisets, *i.e.* they hold an arbitrary number of children including duplicates. An example bigraph used as a (multi-)set is in Fig. 6a.

Checking an entity is in the set is simple: we match on the entity of interest and use a site to allow other children (including the empty child) to be present, *i.e.* the rule Fig. 6b would apply to Fig. 6a. However, matching on the non-existence of a child is difficult. To ensure the entity of interest cannot not appear in a site, we must specify rules for every possible combination of other entities in the set, *e.g.* Fig. 6c. In this case the rule does not apply to Fig. 6a as the circle entity is present. It is often not practical to specify rules for all permutations of additional entities in the set as the number of rules increases factorially.



(a) Example Bigraph

(b) ∃circle - Other entities allowed

(c) ∄circle - No site allowed

(d) ∄circle - Other entities allowed

Fig. 6. Using negative application conditions for non-existence in multisets.

As shown in Fig. 6d, negative application conditions for the parameters allows sites to be used in non-existence checks by allowing anything in the parameter *except* the entity of interest – a concise and natural way of specifying the rule.

A similar issue of sites hiding too much information occurs when counting. For example, if we wish to match at most one T then without negative conditions we must enumerate all possible sets with a single T. With negative conditions, we can specify that a site exists but that the site does not contain more than one T.

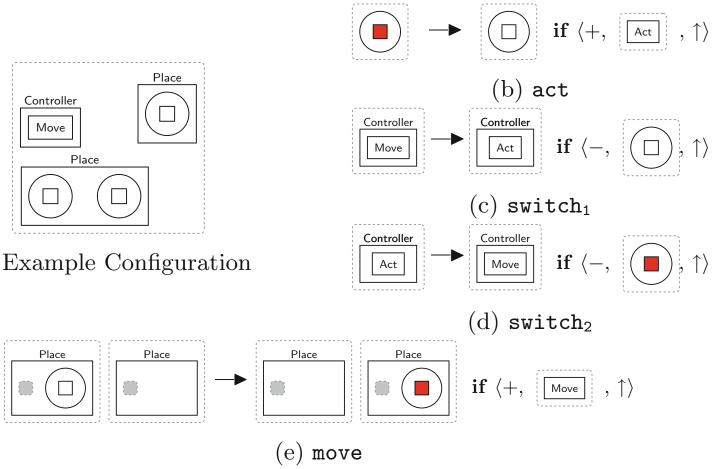
5.3 Encoding Control Flow

Models often need to encode some control flow, for example, to implement turn-based control. Often this is achieved through the use of tagging, counters, and prioritised rules, e.g. [16], that determine when the algorithms should change state.

Application conditions can make it easier to encode elements of control without requiring counting, tagging, or priorities. Consider the system in Fig. 7 that uses turn-based control where entities, shown as circles, representing autonomous agents, cycle between a Move phase and a Act phase, with the current phase determined by a Controller. Each agent keeps track of its local state, *i.e.* what the last action it performed was.

The `move` and `act` rules (Figs. 7b and 7e) show example actions the agents can take. While the `act` shown only changes internal state (represented by the fill color) to say an action has been performed, in practice this would perform some meaningful step. Applications conditions on the rules ensure that agents only perform valid actions for the given controller state. It is possible to write a similar rule without application conditions by matching on Move in a separate region, however this does not allow the agents to be nested *under* a controller and obscures the meaning of the rule.

Rules `switch`$_1$ and `switch`$_2$ (Figs. 7c and 7d) toggle the controller state only once *all* agent has taken the appropriate action, *i.e.* it encodes fix-point behaviour. As the rules simply check for the non-existence of agents waiting to take an action, they work for *any* number of agents without needing to explicitly encode counting. Likewise, there is no need to introduce priorities to the rules as the conditions guard them from firing at the wrong time.



(a) Example Configuration

(b) `act`

(c) `switch`$_1$

(d) `switch`$_2$

(e) `move`

Fig. 7. Encoding turn-based control. (Color figure online)

6 Discussion and Limitations

The key advantage of the presented approach is that application conditions are defined solely in terms of the existing rewriting theory for bigraphs – allowing theory/tool reuse.

However, this approach does not capture all the application conditions we might wish to specify in practice. In this section we highlight the limitations of the current approach and discuss how, by moving away from standard bigraph theory, *i.e.* changing the semantics of matching, or utilising spatial logics, we can express a wider range of application conditions.

6.1 Matching on Names

Commonly, application conditions in GTS make use of graph edges to access the context for a *particular* node, *i.e.* the existence of a link identifies an entity of interest in the context. In bigraphs such an approach is not possible as link names *cannot* be used as identifiers.

For example, consider the reaction rule and application condition in Fig. 8. The intention of the rule is to state that an (empty) circle can be transformed into a red circle when it is connected to a square on the x link. However, this is not the interpretation because an open name in a reaction rule is not an identifier, but, much like sites, it signifies that there *may* be additional entities on the same link. As such, the x on the left-hand side and the x in the application condition are not considered to be the same link (*i.e.* the second x is not bound to the first x); we would get the same rewrite if we replaced, for example, the x in the application condition with y.

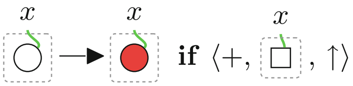


Fig. 8. Names are not identifiers in bigraphs. (Color figure online)

This issue has been observed elsewhere, for example Benford *et al.* [4] extend the matching semantics for bigraphs to "bigraph patterns" that allow matching on specific identifiers.

Utilising this type of matching for application conditions would allow the intended interpretation of Fig. 8, where x is scoped over both the left-hand side and the application condition. No implementation of this matching currently exists, and it remains unclear how this might affect other aspects of the bigraph theory.

6.2 Matches Can Be Too Large

Although we have shown application conditions as defined are useful for practical applications, care must be taken in their use. Currently an application condition allows its constraint to appear *anywhere* in the context/parameter. This is sometimes too strong. Consider our first example of avoiding duplicate links (Fig. 1). If the target bigraph contains an L-L link *within* A, as shown in Fig. 9, then the rule does not apply, even though there is no duplicate link *between* A and B.

Practically such cases are often not an issue as, for example, the `createLink` rule only ever creates links between an A and B – disallowing internal A–A links from ever being created. If internal links are needed, then a different entity type could be used to distinguish between external and internal links. Finally, a *sorting scheme* [15, Chapter 6] could be used to disallow invalid contexts/parameters from existing, however there is currently no automated tool support for checking a sorting scheme is satisfied.

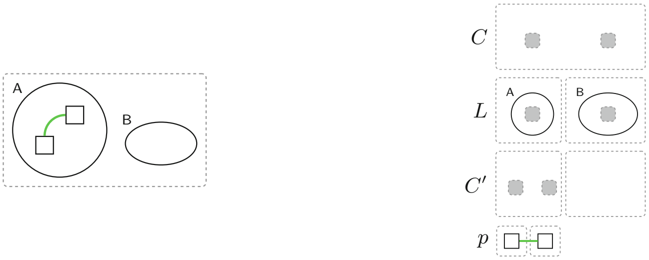


Fig. 9. Matches occur *anywhere* in the parameter, not only in specific regions. Decomposition shows further decomposition $d = C' \circ p$ for application condition constraint p (Fig. 1b) for rule `createLink`.

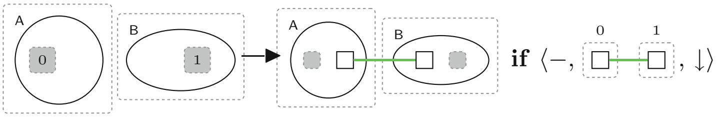


Fig. 10. Explicit parameter placement

6.3 Matching Specific Places

Another possible solution to the internal link issue above is to extend matching to allow application conditions to specify *where* in the context/parameter a particular entity should be found. For example, in Fig. 10, we add explicit indices to the sites allowing us to specify how the application condition should compose with the match – in this case, that the two L ends are in distinct sites. Matching routines that can check for the explicit placement of entities are not currently available, and as with name linking it remains unclear what affect this might have on the rest of the bigraph theory. As it would only be used for application condition matching it could potentially be defined as a special case matching routine. It is particularly unclear how to perform such matches if sharing is allowed *e.g.* as shares can merge regions in the parameter.

Explicit placement is also possible for conditions in the context, this time allowing us to specify that a particular region in the match occurs as a descendent of a site in the context. Figure 11 is an example of this where entity C must be a – not-necessarily direct – descendant of D in the condition.

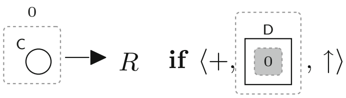
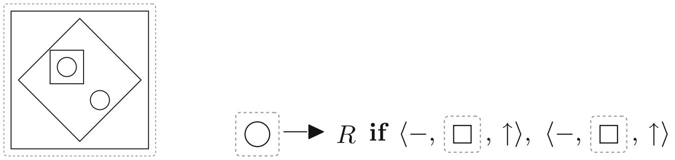


Fig. 11. Explicit context placement

6.4 Handling Overlaps

A conditional rule only applies if the conjunction of all application conditions is true, *e.g.* all matches are present. As each application condition is checked independently, the same entity can be used in multiple matches *i.e.* we allow overlaps between application conditions. For example, consider the rule in Fig. 12. The intended behaviour is to check there are *two* distinct square entities in the context – regardless of how many other entities *e.g.* the diamond, are in the nesting – however as we allow overlaps this matches even in the case a single square entity is in the context.



(a) Example Bigraph (b) Reaction rule with overlapping condition

Fig. 12. Rule applies to both circle entities as overlaps are allowed

A solution for this in GTS is to use *nested conditions* [12] that allow further checks to be made on the context of the constraint *within* in the application condition.

We can define a form of nested application conditions for bigraphs by allowing further decomposition within the application condition match. That is, we first find a suitable context (parameter) and then decompose the context (parameter) into a new context/parameter and check nested conditions on these new context/parameter.

As it only requires additional decompositions, such an approach is possible within the existing bigraph matching framework. However, it requires matches to apply in a specific order *e.g.* closest match first, to ensure we know if the nested condition should be in the parameter or context. For example, in Fig. 12 if we first match on the outermost square then the next condition appears in the parameter. But, if we first match on the innermost square then the next condition is in the context.

6.5 Related Work

Sorting Schemes. It is possible to give a sorting scheme to bigraphs [15, Chapter 6] that determines when a bigraph is well formed, *e.g.* that a Room cannot be within a Person – much like a type graph for GTS. Sorting schemes compliment application conditions. The sorting scheme defines what *can* be in the context/parameter, while application conditions determine what *is* in the current context/parameter. While there is an existing theory for sorts, there is currently no tool support available.

Conditional Transformation Systems. Conditional rewriting is often found in rewriting logic [14] where rules take the form $r \rightarrow l$ if $u_1 \wedge u_2 \wedge \ldots$. Unlike application conditions, rewriting logic allows the terms in u_1 to be reduced over a set of equations.

Application conditions are also a common feature in GTS [10, Chapter 7] and are present in most graph rewriting frameworks. Intuitively, a GTS application condition is defined as the existence/non-existence of a match (graph morphism) between a non-local pattern and the current graph.

More general treatments of (nested) application conditions considers them at the level of adhesive categories [5, 12]. Such categories are are closely linked with the DPO rewriting approach. The relationship between bigraphs and cospan categories/DPO rewriting has previously been explored [9], and such an input-linear variant of bigraphs[5], such as that of Sobociński [21], could fit such a framework.

Spatial Logics. Another approach to application conditions for bigraphs was explored by Tsigkanos *et al.* [22]. Here application conditions are specified using a spatial logic for closure spaces [7], of which graphs are an instance. The logic requires flattening the bigraph to a graph – losing the distinction between spatial and non-spatial links – but provides features such as matching links by name and specifying reachability constraints, *e.g.* a PC connects to a Printer through some path.

A more general spatial logic for bigraphs is BiLog [8], which could also implement application conditions while maintaining the orthogonality between space and linking. However, there is a lack of tool support and the decidability of the logic remains an open question.

Importantly, both logics require the user to specify constraints in a language separate to that of bigraphs, while our approach maintains the diagrammatic approach by having conditions as bigraphs.

7 Conclusion

Reactive modelling formalisms, such as bigraphical reactive systems, should make it as easy as possible to express how a system evolves over time. Whether or not a reaction rule is applicable often depends not only on a local match, but also on the surrounding context. Application conditions allow non-local reasoning to be added to reaction rules allowing the context to be interrogated to check the existence/non-existence of constraints.

We have extended the theory of bigraphical reactive systems with conditional reaction rules that allow application conditions to be specified. Unlike graph transformation systems that feature a single context, bigraphs have both a *context* (above the match) and a *parameter* (below the match). We show how these contexts can be further decomposed as additional instances of the bigraph matching problem, enabling the existing matching framework to be used to check

[5] Standard bigraphs are *output-linear.*

application conditions. To show this is useful in practice we implement conditional rules in BigraphER [19].

Unfortunately, such rules do not let us express all conditions of interest. For example we cannot track a name from the match into the context, or specify the exact location of entities *e.g.* do not apply a rule if entity A is a grandparent. Specifying these types of property require extensions to how bigraphs are matched, and potentially the use of spatial logics to provide exact specification of spatial constraints.

This paper paves the way for future work on application conditions for bigraphs, and, more generally, improvements to the matching algorithm that allow more expressive constraints to be described.

Acknowlegements. This work was supported EPSRC grant S4: Science of Sensor Systems Software (EP/N007565/1), and PETRAS SRF grant MAGIC (EP/S035362/1).

References

1. Alrimawi, F., Pasquale, L., Nuseibeh, B.: On the automated management of security incidents in smart spaces. IEEE Access **7**, 111513–111527 (2019)
2. Archibald, B., Shieh, M., Hu, Y., Sevegnani, M., Lin, Y.: BigraphTalk: verified design of IoT applications. IEEE Internet Things J. **7**(4), 2955–2967 (2020). https://doi.org/10.1109/JIOT.2020.2964026. ISSN 2372-2541
3. Baeten, J.C.M., Bergstra, J.A., Klop, J.W., Weijland, W.P.: Term-rewriting systems with rule priorities. Theor. Comput. Sci. **67**(2&3), 283–301 (1989)
4. Benford, S., Calder, M., Rodden, T., Sevegnani, M.: On lions, impala, and bigraphs: modelling interactions in physical/virtual spaces. ACM Trans. Comput. Hum. Interact. **23**(2), 1–56 (2016)
5. Bruggink, H.J.S., Cauderlier, R., Hülsbusch, M., König, B.: Conditional reactive systems. In: FSTTCS, pp. 191–203 (2011)
6. Calder, M., Koliousis, A., Sevegnani, M., Sventek, J.S.: Real-time verification of wireless home networks using bigraphs with sharing. Sci. Comput. Program. **80**, 288–310 (2014)
7. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and verifying properties of space. In: Diaz, J., Lanese, I., Sangiorgi, D. (eds.) TCS 2014. LNCS, vol. 8705, pp. 222–235. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44602-7_18
8. Conforti, G., Macedonio, D., Sassone, V.: Spatial logics for bigraphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 766–778. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_62
9. Ehrig, H.: Bigraphs meet double pushouts. Bull. EATCS **78**, 72–85 (2002)
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31188-2
11. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inform. **26**(3/4), 287–313 (1996)

12. Habel, A., Pennemann, K.-H.: Nested constraints and application conditions for high-level structures. In: Kreowski, H.-J., Montanari, U., Orejas, F., Rozenberg, G., Taentzer, G. (eds.) Formal Methods in Software and Systems Modeling, Essays Dedicated to Hartmut Ehrig on the Occasion of His 60th Birthday. LNCS, vol. 3393, pp. 293–308. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31847-7_17
13. Krivine, J., Milner, R., Troina, A.: Stochastic bigraphs. Electron. Notes Theor. Comput. Sci. **218**, 73–96 (2008)
14. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theor. Comput. Sci. **96**(1), 73–155 (1992)
15. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press, Cambridge (2009)
16. Muffy, C., Michele, S.: Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. Form. Asp. Comput. **26**(3), 537–561 (2014). https://doi.org/10.1007/s00165-012-0270-3. ISSN 0934-5043
17. Sevegnani, M.: Bigraphs with sharing and applications in wireless networks. Ph.D. thesis, School of Computing Science, University of Glasgow (2012)
18. Sevegnani, M., Calder, M.: Bigraphs with sharing. Theor. Comput. Sci. **577**, 43–73 (2015)
19. Sevegnani, M., Calder, M.: BigraphER: rewriting and analysis engine for bigraphs. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 494–501. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_27
20. Sevegnani, M., Kabác, M., Calder, M., McCann, J.A.: Modelling and verification of large-scale sensor network infrastructures. In: ICECCS, pp. 71–81 (2018)
21. Sobociński, P.: Deriving process congruences from reaction rules. Ph.D. thesis, Aarhus University (2004)
22. Tsigkanos, C., Kehrer, T., Ghezzi, C.: Modeling and verification of evolving cyber-physical spaces. In: ESEC/FSE, pp. 38–48 (2017)